Available online at www.sciencedirect.com

ScienceDirect

Journal of Computational and Applied Mathematics 199 (2007) 372–377

JOURNAL OF COMPUTATIONAL AND APPLIED MATHEMATICS

www.elsevier.com/locate/cam

An efficient and safe framework for solving optimization problems

Yahia Lebbah[a, b], Claude Michel[a, *], Michel Rueher[a]

[a] *Université de Nice–Sophia Antipolis, COPRIN (I3S–CNRS/INRIA/CERTIS), 930 route des Colles, B.P. 145, 06903 Sophia Antipolis, Cedex, France*

[b] *Université d'Oran, Département d'Informatique, 31000 Oran, Algeria*

Received 21 December 2004

Abstract

Interval methods have shown their ability to locate and prove the existence of a global optima in a safe and rigorous way. Unfortunately, these methods are rather slow. Efficient solvers for optimization problems are based on linear relaxations. However, the latter are unsafe, and thus may overestimate, or, worst, underestimate the very global minima. This paper introduces `QuadOpt`, an efficient and safe framework to rigorously bound the global optima as well as its location. `QuadOpt` uses consistency techniques to speed up the initial convergence of the interval narrowing algorithms. A lower bound is computed on a linear relaxation of the constraint system and the objective function. All these computations are based on a safe and rigorous implementation of linear programming techniques. First experimental results are very promising.

Keywords: Global optimization; Safe linear relaxations; Constraint programming

1. Introduction

We consider here the global optimization problem $\mathscr{P}$ to minimize an objective function under nonlinear equalities and inequalities,

$$\begin{aligned} &\text{minimize} && f(x) \\ &\text{subject to} && g_i(x) = 0, \quad i = 1, \ldots, k, \\ &&& g_j(x) \leqslant 0, \quad j = k+1, \ldots, m, \end{aligned} \tag{1}$$

with $x \in \mathbf{x}$, $f : \mathbb{R}^n \to \mathbb{R}$ and $g_j : \mathbb{R}^n \to \mathbb{R}$; Functions f and g_j are continuously differentiable on some vector $\mathbf{x}$ of intervals of $\mathbb{R}$.

Among the many approaches developed to solve optimization problems, two main trends could be distinguished.

The first one, and undoubtedly the most successful one, aims at solving $\mathscr{P}$ in the most efficient fashion. Linear relaxations and local methods are used to speed up the convergence to a global optima. The most famous implementation of this approach is the global optimizer of Sahinidis called `Baron` [14]. However, while fast and *complete* (we use here

* Corresponding author.
E-mail addresses: lebbah@essi.fr (Y. Lebbah), cpjm@essi.fr (C. Michel), rueher@essi.fr (M. Rueher).

0377-0427/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.cam.2005.08.037

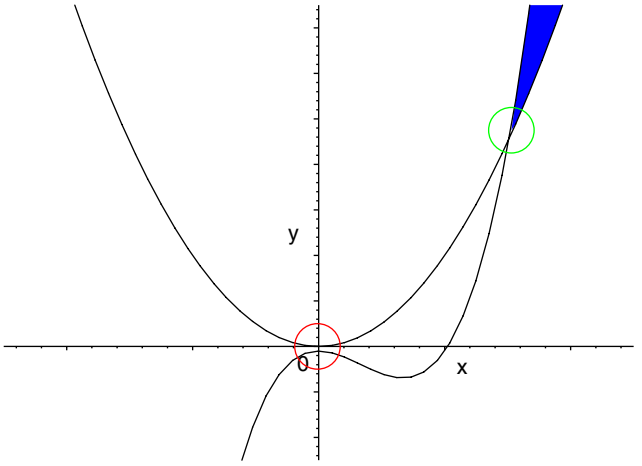

Fig. 1. Geometrical representation of Problem 2.

the classification system proposed in [11]), these methods are not *rigorous*. That is to say, when run on a computer, the result of these algorithms could be an overestimation or, worst, an underestimation of the very global optima.

The second trend mainly relies on interval computation to *rigorously* bound the global optima. The use of outward rounding allows a safe bounding of the global optima by means of a computer. Nevertheless, rigorous systems based on interval computations like the Kearfott's system `Globsol` described in [8] are rather slow.

So, the challenge is to combine the advantages of both approaches in an efficient and rigorous global optimization framework. That is why we propose here to embed safe linear relaxations in an interval- and constraint-based framework.

Before going into details, let us show an a small example a flaw or lack of rigour. Consider the following optimization problem:

$$\begin{aligned} &\text{minimize} && x \\ &\text{subject to} && y - x^2 \geqslant 0, \\ &&& y - x^2(x-2) + 10^{-5} \leqslant 0, \\ &&& x, y \in [-10, +10]. \end{aligned} \tag{2}$$

As shown in Fig. 1, the solution of Problem (2) lies in the neighbourhood of point $x \approx 3$, $y \approx 9$. This point is the unique intersection of curve $y = x^2$ and curve $y = x^2(x-2) - 10^{-5}$. However, at point $x = 0$, $y = 0$, the two curves are only separated by a small distance of 10^{-5}. `Baron` (6.0 and 7.2) quickly finds 0 as the global minimum even if the precision is enforced up to 10^{-12}. Such a flaw is particularly annoying: as pointed out in [11], there are many situations, like safety verification problems or chemistry, where the knowledge of the very global optima is critical.

The rest of this paper is organized as follows. The next section contains the notations. Section 3 gives an overview of the use of safe linear relaxations while Section 4 details our global optimization framework. Section 5 describes first experimental results.

2. Notations

An interval $[\overline{x}, \underline{x}]$ is the set of real numbers x such that $\underline{x} \leqslant x \leqslant \overline{x}$. $\mathbf{x}$, $\mathbf{y}$ denote indifferently intervals and vectors of intervals, also called boxes. If necessary, the text will clearly state whether $\mathbf{x}$ is an interval or a box. The *width* $w(\mathbf{x})$ of an interval $\mathbf{x}$ is the quantity $\overline{x} - \underline{x}$. $\underline{f}^*$ and $\overline{f}^*$, respectively, denote lower and upper bounds of f^*, the optimal value of the objective function f. $\mathbb{R}$ denotes the set of reals while $\mathbb{F}$ denotes a set of floating point numbers.

3. Safe use of linear relaxations

QuadOpt, the new optimization framework we introduce in this paper, is based on the techniques developed in [10] for QuadSolver, a new branch and prune algorithm for handling numerical constraints.

3.1. QuadSolver

QuadSolver uses safe linear relaxations to reduce the domains of the variables. Linear relaxations are combined with local consistencies ($2B$ consistency and Box consistency which are more detailed and compared in [3]) as well as interval methods (e.g., interval Newton) to provide an efficient and safe framework to search the solutions of nonlinear problems.

QuadSolver handles in a global way the constraints by means of the Simplex. Roughly speaking, the approach is based on two steps:

(1) a reformulation step which captures the linear part of the problem: it replaces each nonlinear term by a new variable (e.g. x^2 by y_i);
(2) a linearisation/relaxation step which introduces redundant linear constraints to provide tight linear approximations of the nonlinear terms.

Then, the Simplex algorithm is used to compute $\underline{x}_i = \min x_i \ in$ LP and $\overline{x}_i = \max x_i \ in$ LP, where LP stands for the linear relaxation of the nonlinear problem. More details on QuadSolver could be found in [10].

The point is that most implementations of the simplex algorithm are based on floating point numbers, and thus are unsafe. To get rigorous upper bound of the objective function, QuadSolver implements a simple and cheap procedure which has been introduced in [12].

The coefficient of the generated linear relaxations are computed with floating point numbers and thus, the linearisations may become incorrect due to rounding errors. To overcome this problem QuadSolver, uses a safe procedure when computing the coefficient of the linear relaxations.

In the next section, we give an overview of the rounding process we use to ensure that the linear relaxations are safe.

3.2. Safe linear relaxations

The safe rounding of the linear relaxation coefficients is handled in two complementary ways. The most common and most simple linear relaxations (e.g. x^2) use dedicated procedures to insure the correct rounding of their coefficients. A general procedure to correct any n-ary linearisations is used to handle other linear relaxations.

For example, the nonlinear term x^2 with $\underline{x} \leqslant x \leqslant \overline{x}$ is approximated by

$$L_1(y, \alpha) \equiv y - 2\alpha x + \alpha^2 \geqslant 0, \quad \text{where } \alpha \in [\underline{x}, \overline{x}], \tag{3}$$

$$L_2(y) \equiv (\underline{x} + \overline{x})x - y - \underline{x} * \overline{x} \geqslant 0, \tag{4}$$

where $L_1(y, \alpha)$ generates the tangent to $y = x^2$ at $x = \alpha$. $L_1(y, \alpha)$ underestimates y whereas $L_2(y)$ overestimates y. QuadSolver only computes $L_1(y, \overline{x})$ and $L_1(y, \underline{x})$ which provide a good ratio between the number of linear relaxations and the tightness of the approximation. A rounding error in the computation of the coefficients of $L_1(y, \alpha)$ or $L_2(y)$ could exclude some of the solutions. To avoid the loss of solutions, a safe rounding procedure is applied to the computation of the coefficient of L_1. The following property gives the right rounding direction for the computation of L_1 coefficients:

$$\text{Let } \alpha \in \mathbb{F} \text{ and } L_{1\mathbb{F}}(y, \alpha) \equiv \begin{cases} y - \inf(2\alpha)x + \sup(\alpha^2) \geqslant 0 \ \text{ iff} \alpha \geqslant 0, \\ y - \sup(2\alpha)x + \sup(\alpha^2) \geqslant 0 \ \text{ iff} \alpha < 0. \end{cases}$$

Then for all $x \in \mathbf{x}$, and for all $y \in [0, \max\{\underline{x}^2, \overline{x}^2\}]$, if $L_1(y, \alpha)$ holds, then $L_{1\mathbb{F}}(y, \alpha)$ holds too. Correct rounding for the computation of L_2, as well as the linear relaxation of xy, are detailed in [10].

Some complex linear relaxations like the linearisations generated by the sandwich algorithm—detailed in [13]—are more conveniently handled by a more general approach. Next property sets the right rounding direction for a general n-ary linearisation: Let $\sum_{i=1}^{n} a_i x_i + b \geqslant 0$ then $\forall x_i \in \mathbf{x}_i$.

$$\sum_{i=1}^{n} \overline{a}_i x_i + \sup\left(\overline{b} + \sum_{i=1}^{n} \sup(\sup(\mathbf{a}_i \underline{x}_i) - \overline{a}_i \underline{x}_i)\right) \geqslant \sum_{i=1}^{n} a_i x_i + b \geqslant 0.$$

Note that this generalization is usually less tight than specialized corrections. Borradaile and Van Hentenryck [2] have recently introduced other corrections of n-ary linearisations and Hongthong and Kearfott [6] have recently introduced corrections of other nonlinear terms.

4. From `Quad` to global optimization

`QuadSolver` offers the safe and rigorous tools to build a safe and efficient global optimization framework. That is to say, the rigorous use of linear relaxations from `QuadSolver` can be combined with other classical safe techniques coming from interval methods and constraint programming to prune the feasible space and to compute a safe lower bound.

Our branch and bound algorithm `QuadOpt` combines interval analysis and constraint programming techniques within the well known branch and bound schema described in [7]. Interval analysis techniques enables to introduce safeguards that ensure rigorous and safe computations whereas constraint programming techniques improve the reduction of the feasible space.

`QuadOpt`(see Algorithm 1) computes enclosers for minimizers and safe bounds of the global minimum value within an initial box $\mathbf{x}$. The algorithm maintains two lists: a list $\mathscr{L}$ of boxes to be processed and a list $\mathscr{S}$ of proven feasible boxes. It provides a rigorous encloser $[\underline{f}^*, \overline{f}^*]$ of the global optimum with respect to a given tolerance ε.

Algorithm 1 The `QuadOpt` algorithm

Function `QuadOpt(IN` $\mathbf{x}$, ε; `OUT` $\mathscr{S}$, $[\underline{f}^*, \overline{f}^*]$)
% $\mathscr{S}$: set of proved feasible points
% $\mathbf{f}_{\mathbf{x}}$ denotes the set of possible values for f in $\mathbf{x}$
$\mathscr{L} \leftarrow \{\mathbf{x}\}$; $\mathscr{S} \leftarrow \emptyset$; $(\underline{f}^*, \overline{f}^*) \leftarrow (-\infty, +\infty)$;
  while $w([\underline{f}^*, \overline{f}^*]) > \varepsilon$ **do**
  $\mathbf{x}' \leftarrow \mathbf{x}''$ such that $\underline{\mathbf{f}}_{\mathbf{x}''} = \min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathscr{L}\}$; $\mathscr{L} \leftarrow \mathscr{L} \backslash \mathbf{x}'$;
  $\overline{\mathbf{f}}_{\mathbf{x}'} \leftarrow \min(\overline{\mathbf{f}}_{\mathbf{x}'}, \overline{f}^*)$;
  $\mathbf{x}' \leftarrow Prune(\mathbf{x}')$;
  $\underline{\mathbf{f}}_{\mathbf{x}'} \leftarrow LowerBound(\mathbf{x}')$;
  $(\overline{\mathbf{f}}_{\mathbf{x}'}, \mathbf{x}_p, Proved) \leftarrow UpperBox(\mathbf{x}')$;
  if *Proved* **then** $\mathscr{S} \leftarrow \mathscr{S} \cup \{\mathbf{x}_p\}$; **endif**
  if $\mathbf{x}' \neq \emptyset$ **then** $(\mathbf{x}'_1, \mathbf{x}'_2) \leftarrow Split(\mathbf{x}')$; $\mathscr{L} \leftarrow \mathscr{L} \cup \{\mathbf{x}'_1, \mathbf{x}'_2\}$; **endif**
  if $\mathscr{L} = \emptyset$ **then**
    $(\underline{f}^*, \overline{f}^*) \leftarrow (+\infty, -\infty)$;
  else
    $(\underline{f}^*, \overline{f}^*) \leftarrow (\min\{\underline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathscr{L}\}, \min\{\overline{\mathbf{f}}_{\mathbf{x}''} : \mathbf{x}'' \in \mathscr{S}\})$;
  endif
endwhile

The algorithm selects the box with the lowest lower bound of the objective function. The *Prune* function applies `QuadSolver`'s techniques to reduce the size of the box $\mathbf{x}'$. Then, $LowerBound(\mathbf{x}')$ computes a rigorous lower bound of the objective within the box $\mathbf{x}'$ using `QuadOpt` on a linear programming relaxation of the initial problem. $UpperBox(\mathbf{x})$ computes a feasible box. A local search method helps to quickly find an approximate feasible point. Interval techniques are used to check the feasibility of the provided box (We rely on the techniques introduced in [4] to

		QuadOpt			Globsol			Baron		
Name	*(n,m)*	*Safe*	*T(s)*	*Splits*	*Safe*	T(s)	*Splits*	*Safe*	*T(s)*	*Splits*
TP16	(2,2)	*	0.02	0	*	0.03	–	?	0.02	–
TP220	(2,1)	*	0.01	0	*	0.06	–	?	0.00	–
TP265	(4,2)	*	0.09	2	–	8.51	–	?	0.02	–
TP33	(3,2)	*	0.07	0	*	0.08	4	?	0.03	–
TP55	(6,6)	*	0.07	0	–	1.64	–	?	0.02	–
Audet140a	(5,4)	*	0.15	1	*	4.50	974	?	0.06	–
Audet140b	(4,2)	*	0.07	0	*	0.17	–	?	0.04	–
Audet141	(6,4)	*	0.31	1	*	2.52	57	?	0.12	–
Audet145	(7,8)	*	0.26	0	*	48.57	427	?	0.10	–
Audet146	(10,12)	*	0.80	0	–	3635.73	?	?	0.46	–
Audet147	(16,19)	*	0.54	0	*	∞	?	?	0.16	–
Audet149	(10,24)	*	546.12	363	–	∞	?	?	3.66	–

Fig. 2. Running `QuadOpt`, `Globsol` and `Baron` on some benches.

handle under-determined systems). If *UpperBox* succeeds to prove feasibility then the box $\mathbf{x}_p$ that contains this proven feasible point is added to the list $\mathscr{S}$. At this stage, if the box $\mathbf{x}'$ is empty then, either it does not contain any feasible point or its lower bound $\underline{\mathbf{f}}_{\mathbf{x}'}$ is greater than the current upper bound $\overline{f}^*$. In both cases, we say that the box is fathomed. If $\mathbf{x}'$ is not empty, the box is split along one of the problem variables.[1] At each box selection and processing, the algorithm maintains the lowest lower bound $\underline{f}^*$ of the remaining boxes $\mathscr{L}$ and the lowest upper bound $\overline{f}^*$ of proven feasible boxes. The algorithm terminates when the space between $\overline{f}^*$ and $\underline{f}^*$ becomes smaller than the given tolerance ε. Of course a proven optimum cannot always be found, and thus, algorithm 1 has to be stopped in some cases to get the feasible boxes which may have been found.

5. Experimentations

This section compares the results obtained on some well-known benches with `QuadOpt`, `Globsol` and `Baron`. The TPs problems come from the benches proposed in [5], while Audet's problems come from his thesis [1]. All the tests have been run on a laptop with a Pentium III at 1.2 Ghz. `QuadOpt` uses Ilog Cplex to solve linear problems and IpOpt to search for a local optima.

Fig. 2 presents the results of our experimentations. In this figure, n is the number of variables and m is the number of constraints; $T(s)$ is the time in second required to solve the problem and Splits is the number of splits.

These benches show that `QuadOpt` is almost always faster than `Globsol` and compares well with `Baron`. In [9], Kearfott describes a new version of `Globsol` which tries to take advantage of safe linear relaxations. `QuadOpt` outperforms this version: `QuadOpt` requires only 59.55 s to solve ex5.2.4, 109.04 s to solve ex8.1.7, 0.27 s to solve ex9.2.4 and 2.58 s to solve ex9.2.5 whereas `Globsol` needs more than one hour to solve each of these benches on a faster computer.

6. Conclusion and future works

In this paper, we have introduced a new safe and efficient framework to compute the global optima of a nonlinear problem. Though the first results are promising, some observations have shown that we still have room for improvement; especially for the computation of the lower bound.

[1] Various are heuristics are used to select the variable the domain of which has to be split.

References

[1] C. Audet, Optimisation globale structurée: Propriétés, équivalences et résolution, Ph.D. Thesis, École Polythecnique de Montréal, Montréal, Quebec, 1997.
[2] G. Borradaile, P.V. Hentenryck, Safe and tight linear estimators for global optimization, Math. Programming 102 (3) (2005) 495–517.
[3] H. Collavizza, F. Delobel, M. Rueher, Comparing partial consistencies, Reliab. Comput. 5 (3) (1999) 213–228.
[4] E. R. Hansen, Global Optimization Using Interval Analysis. Marcel Dekker, New York, 2004.
[5] W. Hock, K. Schittkowski, A comparative performance evaluation of 27 nonlinear programming codes, Computing (1983) 335–358.
[6] S. Hongthong, R.B. Kearfott, Rigorous linear overestimators and underestimators, Math. Programming (submitted).
[7] R. Horst, H. Tuy, Global Optimization: Deterministic Approaches, Springer, New York, 1993.
[8] R.B. Kearfott, Rigorous Global Search: Continuous Problems, Kluwer, Dordrecht, 1996.
[9] R. B. Kearfott, Discussion and empirical comparisons of linear relaxations and alternate techniques in validated deterministic global optimization, Optimization Methods and Software (submitted).
[10] Y. Lebbah, C. Michel, M. Rueher, J. Merlet, D. Daney, Efficient and safe global constraints for handling numerical constraint systems, SIAM J. Numer. Anal. 42 (5) (2005) 2076–2097.
[11] A. Neumaier, Complete search in continuous global optimization and constraint satisfaction, Acta Numerica.
[12] A. Neumaier, O. Shcherbina, Safe bounds in linear and mixed-integer programming, in Math. Programming A., DOI 10.1007/s10107-003-0433-3.
[13] G. Rote, The convergence rate of the sandwich algorithm for approximating convex functions, Computing (1992) 337–361.
[14] V. Sahinidis, M. Twarmalani, Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming, Kluwer, Dordrecht, 2002.